



## COLLEGE OF COMPUTING AND INFORMATION SCIENCES

### Final Assessment of Lab Exam (Fall 2020 Semester)

Class Id	105032	Course Title	Numerical Computing
Program	BSCS	Campus / Shift	Main Campus, Morning
Date	23-11-2020	Total Marks	20
Duration	03 hours	Faculty Name	Muhammad Waqas Malik
Student Id	7791	Student Name	Muhammad Umair
Code			

#### Instructions:

- Fill out your Student ID and Student Name in above header.
- Do not remove or change any part question paper.
- Write down your answers with title "Answer# Question# 00".
- Handwritten text or image should be on A4 size page with clear visibility of contents.
- In case of CHEATING, COPIED material or any unfair means would result in negative marking or ZERO.
- Each question carry equal marks.
- You need to print your name and id at the start of each code using print statement.
- **Caution:** Duration to perform Final Assessment is **02 hours only and 01 hour** is given to cater all kinds of odds in .  
**failed to upload answer sheet on LMS (in PDF format) within 3 hours limit, you would be considered as ABSENT**

Fill the table with appropriate values			
Enter your Student ID in brackets		R = [7791]	
R[0]	R[2]	R[-1]	R[-2]
<u>  7  </u>	<u>  9  </u>	<u>  1  </u>	<u>  9  </u>

## Question # 1:

Apply **Bisection Method** to given data and find the root.

$a = R[-1] + 1$

$b = R[-1] + 5$

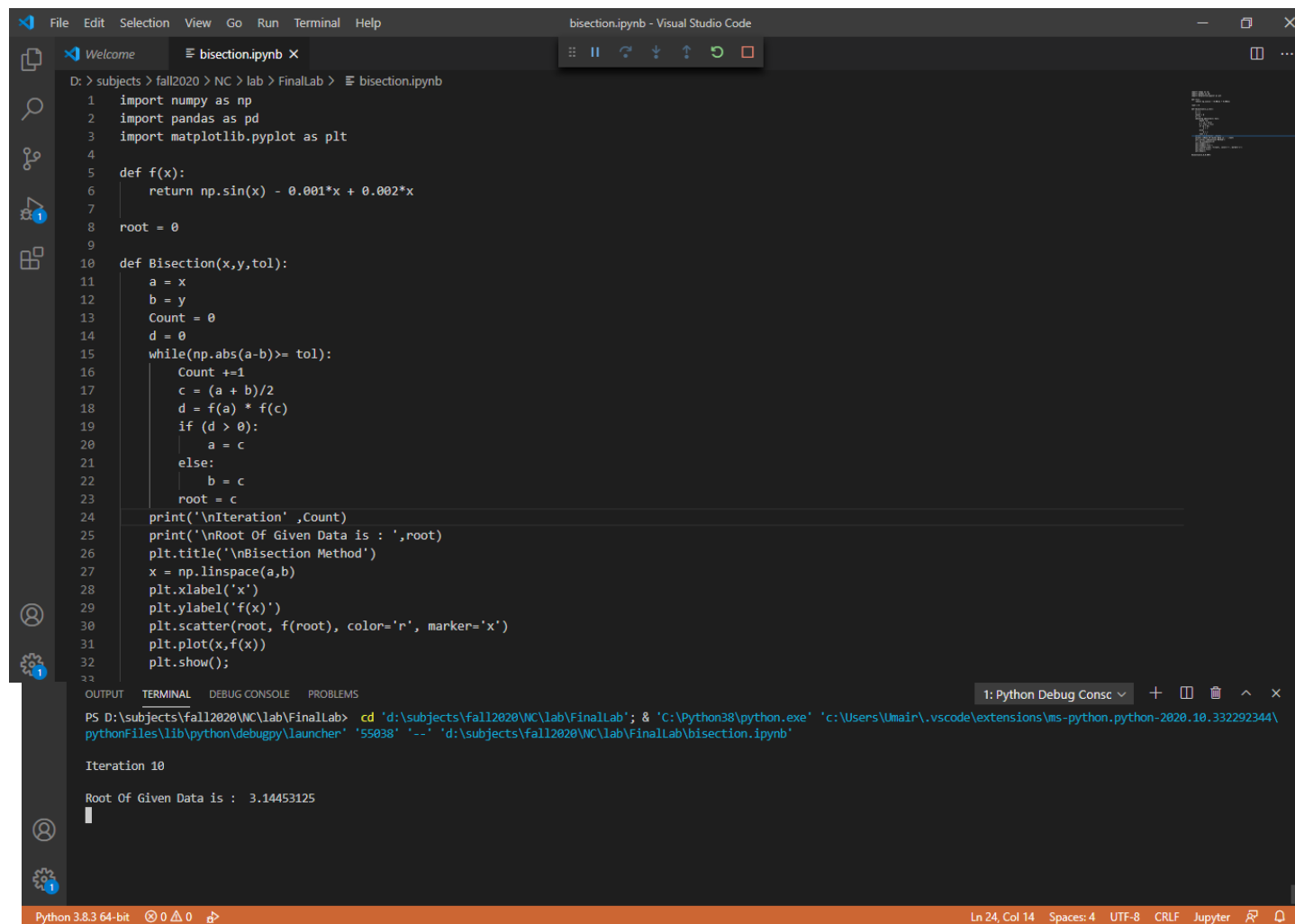
$\text{tol} = 0.00R[0]$

$f(x) = \sin(x) - 0.001x + 0.002x$

$a = 1 + 1 \Rightarrow A = 2$

$b = 1 + 5 \Rightarrow B = 6$

$\text{tol} = 0.007$



```
File Edit Selection View Go Run Terminal Help
bisection.ipynb - Visual Studio Code

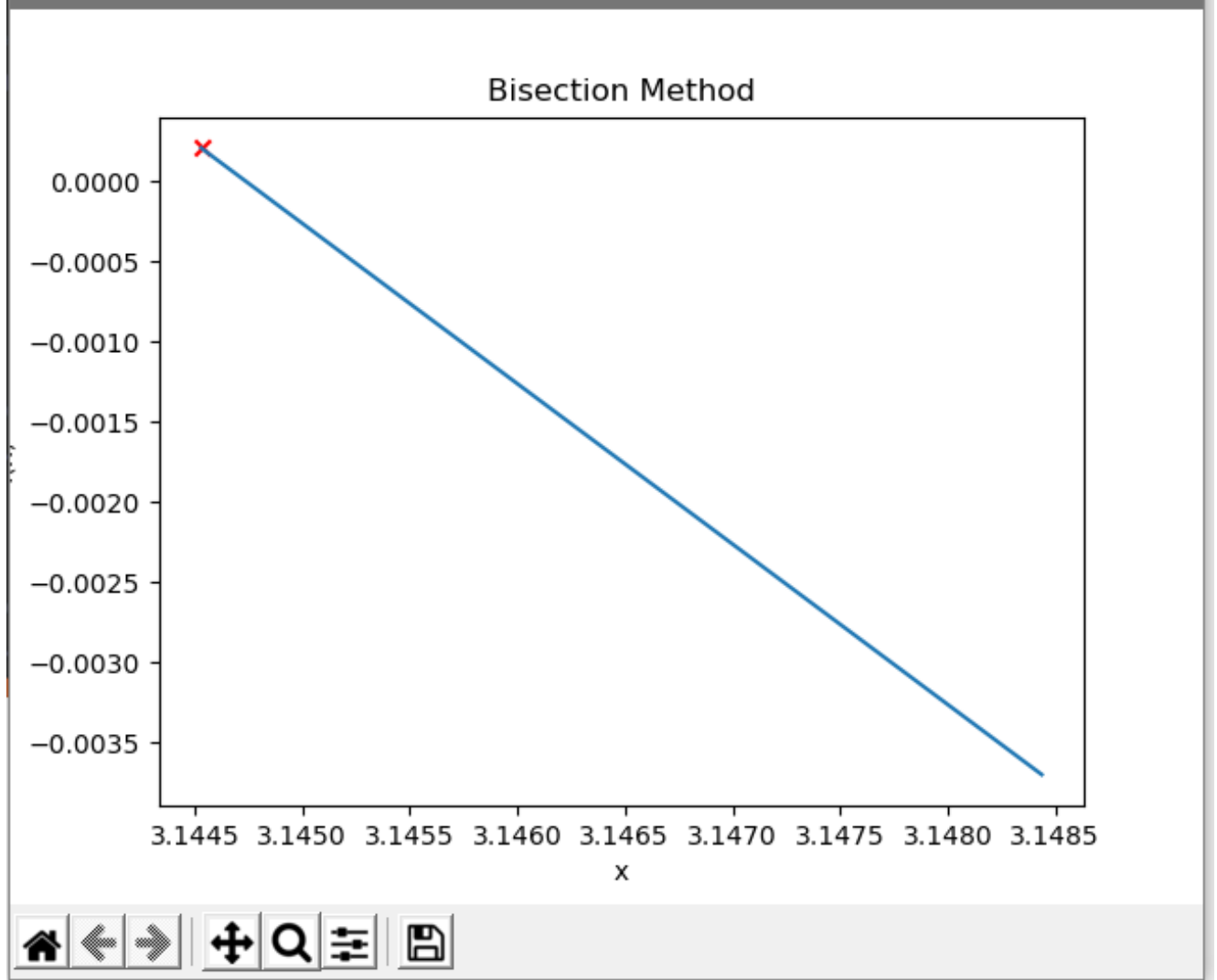
Welcome
bisection.ipynb X

D:\> subjects > fall2020 > NC > lab > FinalLab > bisection.ipynb
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 def f(x):
6     return np.sin(x) - 0.001*x + 0.002*x
7
8 root = 0
9
10 def Bisection(x,y,tol):
11     a = x
12     b = y
13     Count = 0
14     d = 0
15     while(np.abs(a-b)>= tol):
16         Count +=1
17         c = (a + b)/2
18         d = f(a) * f(c)
19         if (d > 0):
20             a = c
21         else:
22             b = c
23         root = c
24     print('\nIteration' ,Count)
25     print('\nRoot Of Given Data is : ',root)
26     plt.title('\nBisection Method')
27     x = np.linspace(a,b)
28     plt.xlabel('x')
29     plt.ylabel('f(x)')
30     plt.scatter(root, f(root), color='r', marker='x')
31     plt.plot(x,f(x))
32     plt.show();
33

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS
1: Python Debug Consc + [ ] [ ] ^ x
PS D:\subjects\fall2020\NC\lab\FinalLab> cd 'd:\subjects\fall2020\NC\lab\FinalLab'; & 'C:\Python38\python.exe' 'c:\Users\Umain\.vscode\extensions\ms-python.python-2020.10.332292344\pythonFiles\lib\python\debugpy\launcher' '55038' '--' 'd:\subjects\fall2020\NC\lab\FinalLab\bisection.ipynb'

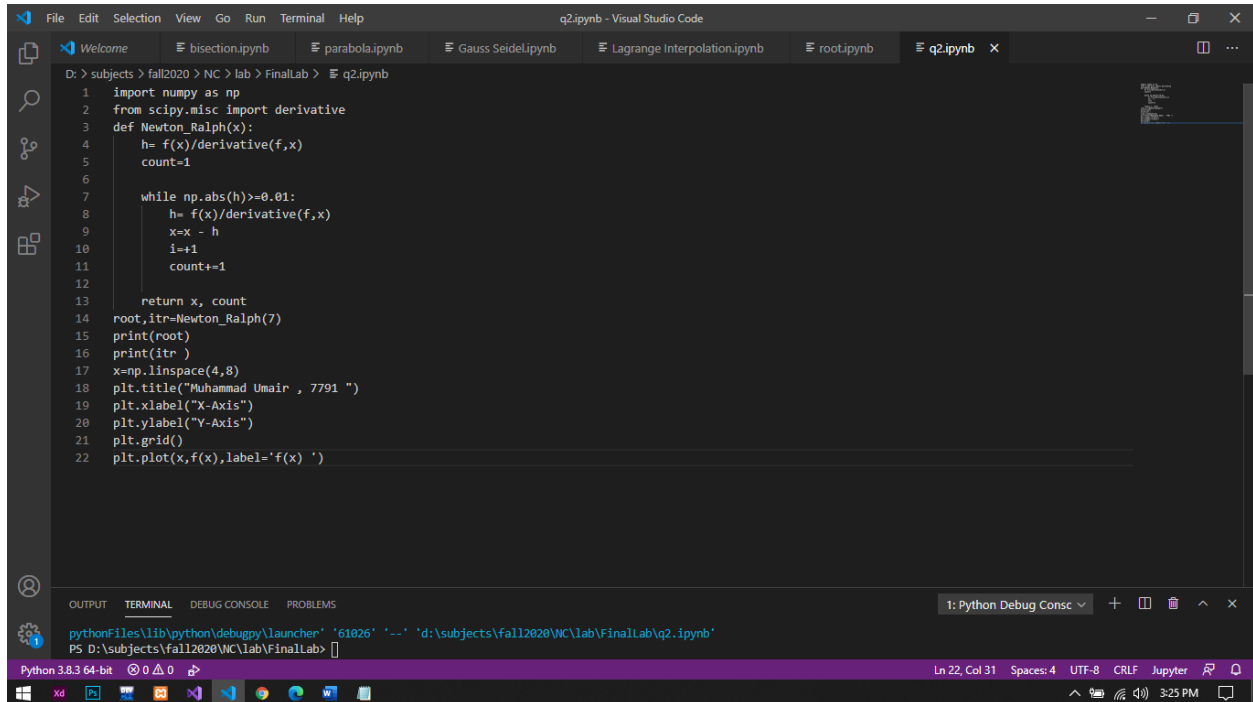
Iteration 10
Root Of Given Data is : 3.14453125
|
```

Figure 1



## Question # 2:

Assume you have a parabola, apply **convergence** on it for  $R[0] + R[-1]$  iterations. Your initial guess is  $R[-1] + 5$  also plot the graph using matplotlib.



```
D:\> subjects > fall2020 > NC > lab > FinalLab > q2.ipynb
1  import numpy as np
2  from scipy.misc import derivative
3  def Newton_Ralph(x):
4      h= f(x)/derivative(f,x)
5      count=1
6
7      while np.abs(h)>=0.01:
8          h= f(x)/derivative(f,x)
9          x=x - h
10         i+=1
11         count+=1
12
13     return x, count
14 root,itr=Newton_Ralph(7)
15 print(root)
16 print(itr )
17 x=np.linspace(4,8)
18 plt.title("Muhammad Umair , 7791 ")
19 plt.xlabel("X-Axis")
20 plt.ylabel("Y-Axis")
21 plt.grid()
22 plt.plot(x,f(x),label='f(x) ')
```

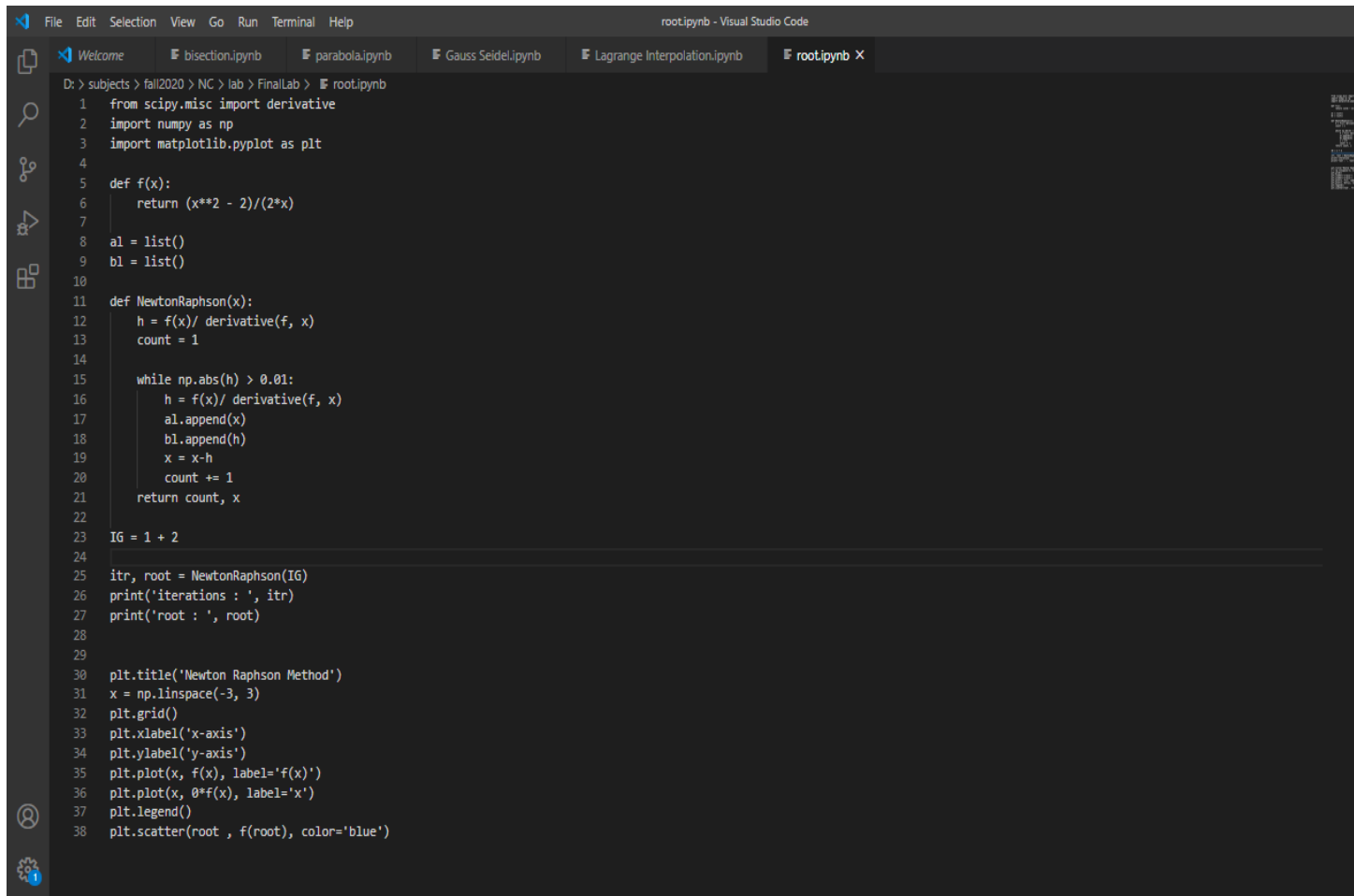
pythonFiles\lib\python\debugpy\launcher '61026' '-' 'd:\subjects\fall2020\NC\lab\FinalLab\q2.ipynb'  
PS D:\subjects\fall2020\NC\lab\FinalLab>

## Question # 3:

Find the **root** however 1<sup>st</sup> step is given below.

$$x_{i+1} = x_i - \frac{x_i^2 - 2}{2x_i}$$

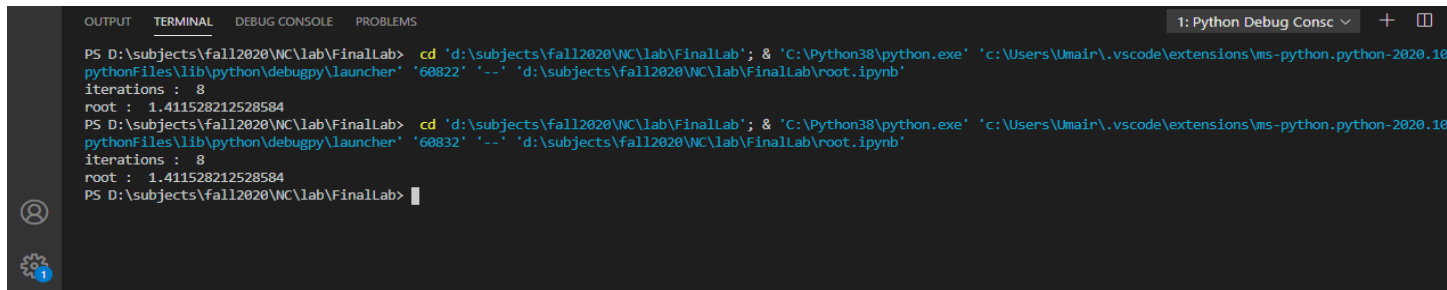
Initial Guess =  $R[-1] + 2$



```
File Edit Selection View Go Run Terminal Help
root.ipynb - Visual Studio Code

Welcome | bisection.ipynb | parabola.ipynb | Gauss Seidel.ipynb | Lagrange Interpolation.ipynb | root.ipynb X

D:\> subjects > fall2020 > NC > lab > FinalLab > root.ipynb
1 from scipy.misc import derivative
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 def f(x):
6     return (x**2 - 2)/(2*x)
7
8 al = list()
9 bl = list()
10
11 def NewtonRaphson(x):
12     h = f(x)/ derivative(f, x)
13     count = 1
14
15     while np.abs(h) > 0.01:
16         h = f(x)/ derivative(f, x)
17         al.append(x)
18         bl.append(h)
19         x = x-h
20         count += 1
21     return count, x
22
23 IG = 1 + 2
24
25 itr, root = NewtonRaphson(IG)
26 print('iterations : ', itr)
27 print('root : ', root)
28
29
30 plt.title('Newton Raphson Method')
31 x = np.linspace(-3, 3)
32 plt.grid()
33 plt.xlabel('x-axis')
34 plt.ylabel('y-axis')
35 plt.plot(x, f(x), label='f(x)')
36 plt.plot(x, 0*f(x), label='x')
37 plt.legend()
38 plt.scatter(root, f(root), color='blue')
```



```
OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 1: Python Debug Consc v + □
PS D:\subjects\fall2020\NC\lab\FinalLab> cd 'd:\subjects\fall2020\NC\lab\FinalLab'; & 'C:\Python38\python.exe' 'c:\Users\Umar\.vscode\extensions\ms-python.python-2020.10
pythonFiles\lib\python\debugpy\launcher' '60822' '--' 'd:\subjects\fall2020\NC\lab\FinalLab\root.ipynb'
iterations : 8
root : 1.411528212528584
PS D:\subjects\fall2020\NC\lab\FinalLab> cd 'd:\subjects\fall2020\NC\lab\FinalLab'; & 'C:\Python38\python.exe' 'c:\Users\Umar\.vscode\extensions\ms-python.python-2020.10
pythonFiles\lib\python\debugpy\launcher' '60832' '--' 'd:\subjects\fall2020\NC\lab\FinalLab\root.ipynb'
iterations : 8
root : 1.411528212528584
PS D:\subjects\fall2020\NC\lab\FinalLab>
```

Question # 4:

Use the **Gauss Seidel** Method with relaxation to solve  $Ax=b$ , where

$$\begin{bmatrix} R[0] & -1 & 0 & 0 \\ -1 & R[0] & -1 & 0 \\ 0 & -1 & R[0] & -1 \\ 0 & 0 & -1 & R[0] \end{bmatrix}, b = \begin{bmatrix} 15 \\ 10 \\ 10 \\ 10 \end{bmatrix},$$

AN.UMAR, 7791

Q<sub>NO4</sub>:

$$\begin{bmatrix} R[0] & -1 & 0 & 0 \\ -1 & R[0] & -1 & 0 \\ 0 & -1 & R[0] & -1 \\ 0 & 0 & -1 & R[0] \end{bmatrix} \begin{bmatrix} 15 \\ 10 \\ 10 \\ 10 \end{bmatrix}$$
$$\begin{bmatrix} 7 & -1 & 0 & 0 \\ -1 & 7 & -1 & 0 \\ 0 & -1 & 7 & -1 \\ 0 & 0 & -1 & 7 \end{bmatrix} \begin{bmatrix} 15 \\ 10 \\ 10 \\ 10 \end{bmatrix}$$

The screenshot shows a Visual Studio Code window with a Python file named 'Gauss Seidel.ipynb'. The code implements the Gauss-Seidel method for solving a system of linear equations. The terminal output shows the results of the iteration for the first four values of x.

```

1 import numpy as np
2 import pandas as pd
3
4 df = pd.DataFrame(columns=['x1', 'x2', 'x3', 'x4'])
5 c = np.array([[7, -1, 0, 0, 15], [-1, 7, -1, 0, 10], [0, -1, 7, -1, 10], [0, 0, -1, 7, 10]])
6
7 x = np.zeros(len(c))
8 for k in range(0, 9):
9     for i in range(0, len(c)):
10         y = 0
11         for j in range(0, len(c)):
12             if i != j:
13                 y = y + (c[i][j] * x[j])
14         y = c[i][len(c)] - y
15         x[i] = y / c[i][i]
16 for i in range(len(x)):
17     print("\nX", (i+1), " = ", x[i])

```

Terminal Output:

```

PS D:\subjects\fall2020\WC\lab\FinalLab> cd 'd:\subjects\fall2020\WC\lab\FinalLab'; & 'C:\Python38\python.exe' 'c:\Users\Umar\vscode\extensions\ms-python.python-2020.10.33229\pythonFiles\lib\python\debugpy\launcher' '60559' '--' 'd:\subjects\fall2020\WC\lab\FinalLab\Gauss Seidel.ipynb'

X 1 = 2.4368070951664325
X 2 = 2.057649667339473
X 3 = 1.966740576481351
X 4 = 1.7095343680687645
PS D:\subjects\fall2020\WC\lab\FinalLab>

```

### Question # 5:

First fill the given table with appropriate values and apply **Lagrange Interpolation** for the given data.

x	$R[-1] - 5 =$	$R[-1] - 2 =$	$R[-1] =$	$R[-1] + 2 =$	$R[-1] + 5 =$	$R[-1] + 7 =$
Y	$2 * x - 1 =$ _____	$2 * x - 0 =$ _____	$2 * x - 1 =$ _____	$2 * x - 0 =$ _____	$2 * x - 1 =$ _____	$2 * x - 1 =$ _____

Find the value of

i)  $x_p = R[-1] + 4$     ii)  $x_p = R[-1] - 3$

Q.No 5:

$R[-1] = 1$

S-Id: 7791

$X$	$Y$
$1 - 5 = -4 = x_0$	$2 \times (-4) - 1 = -9$
$1 - 2 = -1 = x_1$	$2 \times (-1) - 1 = -3$
$1 = 1 = x_2$	$2 \times 1 - 1 = 1$
$1 + 2 = 3 = x_3$	$2 \times 3 - 1 = 5$
$1 + 5 = 6 = x_4$	$2 \times 6 - 1 = 11$
$1 + 7 = 8 = x_5$	$2 \times 8 - 1 = 15$

i)  $x_p = 1 + 4 = 5$

ii)  $x_p = 1 - 3 = -2$



File Edit Selection View Go Run Terminal Help

Lagrange Interpolation.ipynb - Visual Studio Code

Welcome

bisection.ipynb

parabola.ipynb

Gauss Seidel.ipynb

Lagrange Interpolation.ipynb X

D: > subjects > fall2020 > NC > lab > FinalLab > Lagrange Interpolation.ipynb

```
1 import numpy as np
2
3 n = 6
4
5 x = [-4,-1,1,3,6,8]
6 y = [-9,-3,1,5,11,15]
7
8 for i in range(2):
9     xp = float(input('\nEnter interpolation point '+str((i+1))+' : '))
10    yp = 0
11    for i in range(n):
12        k = 1
13        for j in range(n):
14            if(i != j):
15                k = k * ((xp - x[j])/(x[i] - x[j]))
16        yp = yp + k * y[i]
17
18    print('\nInterpolated values at %.3f is %.3f'%(xp,yp))
```

OUTPUT

TERMINAL

DEBUG CONSOLE

PROBLEMS

1: Python Debug Consc

```
PS D:\subjects\fall2020\WC\lab\FinalLab> cd 'd:\subjects\fall2020\WC\lab\FinalLab'; & 'C:\Python38\python.exe' 'c:\Users\Umainr\.vscode\extensions\ms-python.python-2020.10.1\pythonFiles\lib\python\debugpy\launcher' '60642' '--' 'd:\subjects\fall2020\WC\lab\FinalLab\Lagrange Interpolation.ipynb'

Enter interpolation point 1 : 5

Interpolated values at 5.000 is 9.000

Enter interpolation point 2 : -2

Interpolated values at -2.000 is -5.000
PS D:\subjects\fall2020\WC\lab\FinalLab>
```

